Finding all maximal perfect haplotype blocks in linear time

Alanko, J.

2020-02-10

Alanko , J , Bannai , H , Cazaux , B , Peterlongo , P & Stoye , J 2020 , ' Finding all maximal perfect haplotype blocks in linear time ' , Algorithms for Molecular Biology , vol. 15 , no. 1 , 2 . https://doi.org/10.1186/s13015-020-0163-6

http://hdl.handle.net/10138/313160
https://doi.org/10.1186/s13015-020-0163-6

Alanko *et al. Algorithms Mol Biol* (2020) 15:2
https://doi.org/10.1186/s13015-020-0163-6

Algorithms for Molecular Biology

RESEARCH

Open Access

Finding all maximal perfect haplotype blocks in linear time

Jarno Alanko[1], Hideo Bannai[2], Bastien Cazaux[1], Pierre Peterlongo[3] and Jens Stoye[4*]

Abstract

Recent large-scale community sequencing efforts allow at an unprecedented level of detail the identification of genomic regions that show signatures of natural selection. Traditional methods for identifying such regions from individuals' haplotype data, however, require excessive computing times and therefore are not applicable to current datasets. In 2019, Cunha et al. (Advances in bioinformatics and computational biology: 11th Brazilian symposium on bioinformatics, BSB 2018, Niterói, Brazil, October 30 - November 1, 2018, Proceedings, 2018. https://doi.org/10.1007/978-3-030-01722-4_3) suggested the *maximal perfect haplotype block* as a very simple combinatorial pattern, forming the basis of a new method to perform rapid genome-wide selection scans. The algorithm they presented for identifying these blocks, however, had a worst-case running time quadratic in the genome length. It was posed as an open problem whether an optimal, linear-time algorithm exists. In this paper we give two algorithms that achieve this time bound, one conceptually very simple one using suffix trees and a second one using the positional Burrows–Wheeler Transform, that is very efficient also in practice.

Keywords: Population genomics, Selection coefficient, Haplotype block, Positional Burrows–Wheeler Transform

Introduction and background

As a result of the technological advances that went hand in hand with the genomics efforts of the last decades, today it is possible to experimentally obtain and study the genomes of large numbers of individuals, or even multiple samples from an individual. For instance, the *National Human Genome Research Institute* and the *European Bioinformatics Institute* have collected more than 3500 genome-wide association study publications in their *GWAS Catalog* [1].

Probably the most prominent example of large-scale sequencing projects is the *1000 Genomes Project* (now *International Genome Sample Resource*, IGSR), initiated with the goal of sequencing the genomes of more than one thousand human individuals to identify 95% of all genomic variants in the population with allele frequency of at least 1% (down toward 0.1% in coding regions). The final publications from phase 3 of the project report about genetic variations from more than 2500 genomes [2, 3].

Recently, several countries announced large-scale national research programs to capture the diversity of their populations, while some of these efforts started already more than 20 years ago. Since 1996 Iceland's deCODE company is mining Icelanders' genetic and medical data for disease genes. In 2015, deCODE published insights gained from sequencing the whole genomes of 2636 Icelanders [4]. *Genome of the Netherlands* (GoNL) is a whole genome sequencing project aiming to characterize DNA sequence variation in the Dutch population using a representative sample consisting of 250 trio families from all provinces in the Netherlands. In 2016, GoNL analysed whole genome sequencing data of 769 individuals and published a haplotype-resolved map of 1.9 million genome variants [5]. Similar projects have been established in larger scale in the UK: Following the *UK10K* project for identifying rare genetic variants

*Correspondence: jens.stoye@uni-bielefeld.de
[4] Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Bielefeld, Germany
Full list of author information is available at the end of the article

in health and disease (2010–2013), *Genomics England* was set up in late 2012 to deliver the 100,000 Genomes Project [6]. This flagship project has by now sequenced 100,000 whole genomes from patients and their families, focusing on rare diseases, some common types of cancer, and infectious diseases. The scale of these projects is culminating in the US federal *Precision Medicine Initiative*, where the NIH is funding the *All of Us* research program[1] to analyze genetic information from more than 1 million American volunteers. Even more extreme suggestions go as far as to propose "to sequence the DNA of all life on Earth"[2].

The main motivation for the collection of these large and comprehensive data sets is the hope for a better understanding of genomic variation and how variants relate to health and disease, but basic research in evolution, population genetics, functional genomics and studies on demographic history can also profit enormously.

One important approach connecting evolution and functional genomics is the search for genomic regions under natural selection based on population data. The *selection coefficient* [7] is an established parameter quantifying the relative fitness of two genetic variants. Unfortunately, haplotype-based methods for estimating selection coefficients have not been designed with the massive genome data sets available today in mind, and may therefore take prohibitively long when applied to large-scale population data. In view of the large population sequencing efforts described above, methods are needed that—at similar sensitivity—scale to much higher dimensions.

Only recently a method for the fast computation of a genome-wide selection scan has been proposed that can be computed quickly even for large datasets [8]. The method is based on a very simple combinatorial string pattern, *maximal perfect haplotype blocks*. Although considerably faster than previous methods, the running time of the algorithm presented in that paper is not optimal, as it takes $O(kn^2)$ time in order to find all maximal perfect haplotype blocks in k genomes of length n each. This is sufficient to analyse individual human chromosomes on a laptop computer, for datasets of the size of the 1000 Genomes Project (thousands of genomes and millions of variations). However, with the larger datasets currently underway and with higher resolution it will not scale favourably. More efficient methods are therefore necessary and it was phrased as an open question whether there exists a linear-time algorithm to find all maximal perfect haplotype blocks.

In this paper we settle this open problem affirmatively. More specifically, after some basic definitions in "Basic definitions" section we present in "Linear-time method I: based on suffix trees" and "Linear-time method II: based on the positional BWT" sections two new algorithms for finding all maximal perfect haplotype blocks in optimal time. The latter of these two algorithms is then experimentally compared to the one from [8] in "Empirical evaluation" section, proving its superiority in running time by a factor of about 5 and memory usage by up to two orders of magnitude for larger data sets. "Conclusion" section concludes the paper.

This paper is an extended version of the preliminary work presented in [9]. Source code and test data are available from https://gitlab.com/bacazaux/haploblocks.

Basic definitions

The typical input to genome-wide selection studies is a set of haplotype-resolved genomes, or *haplotypes* for short. Clearly, for a given set of haplotypes only those sites are of interest where there is variation in the genomes. Therefore, formally, we consider as input to our methods a $k \times n$ *haplotype matrix* where each of the k rows corresponds to one haplotype and each of the n columns corresponds to one variable genetic site.

Most methods distinguish only between ancestral and derived allele, reflecting the fact that most sites are biallelic. Therefore the entries in a haplotype matrix are often considered binary where the ancestral allele is encoded by 0 and the derived allele is encoded by 1. However, the computational problem and its solutions considered in this paper do not depend on this restriction and instead are applicable to any type of sequence over a constant-size alphabet Σ.

The concept of a maximal perfect haplotype block as defined in [8] is the following, where $s[i, j]$ denotes the substring of a string s from position i to position j and $S|_K$ denotes the elements of an ordered set S restricted to index set K:

Definition 1 Given k sequences $S = (s_1, \ldots, s_k)$ of the same length n (representing the rows of a haplotype matrix), a *maximal perfect haplotype block* is a triple (K, i, j) with $K \subseteq \{1, \ldots, k\}$, $|K| \geq 2$ and $1 \leq i \leq j \leq n$ such that

1. $s[i,j] = t[i,j]$ for all $s, t \in S|_K$ *(equality)*,
2. $i = 1$ or $s[i-1] \neq t[i-1]$ for some $s, t \in S|_K$ *(left-maximality)*,
3. $j = n$ or $s[j+1] \neq t[j+1]$ for some $s, t \in S|_K$ *(right-maximality)*, and

[1] http://www.allofus.nih.gov.

[2] Biologists propose to sequence the DNA of all life on Earth, by Elizabeth Pennisi. Science News, Feb. 24, 2017. https://doi.org/10.1126/science.aal0824.

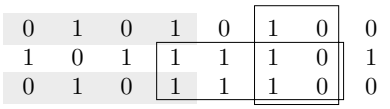

Fig. 1 Illustration of Definition 1. A binary 3 × 8 haplotype matrix with three maximal perfect haplotype blocks ($\{1,3\},1,4$), ($\{2,3\},4,7$) and ($\{1,2,3\},6,7$) highlighted. (The example contains additional maximal perfect haplotype blocks that are not shown.)

4 $\nexists K' \subseteq \{1,\ldots,k\}$ with $K \subset K'$ such that $s[i,j] = t[i,j]$ for all $s,t \in S|_{K'}$ *(row-maximality).*

Definition 1 is illustrated in Fig. 1.

In Cunha et al. [8] it was shown that the number of maximal perfect haplotype blocks is $O(kn)$, while the algorithm presented there takes $O(kn^2)$ time to find all blocks. It is based on the observation that branching vertices in the trie T_p of the suffixes of the input sequences starting at position p correspond to right-maximal and row-maximal blocks, while left-maximality can be tested by comparing T_p and T_{p-1}. In the next two sections we show how this running time can be improved.

Linear-time method I: based on suffix trees

In this section, we present our first algorithm to find all maximal perfect haplotype blocks in linear time. This solution is purely theoretical, it would likely require large amounts of memory while being slow in practice. However, it demonstrates the connection to the concept of maximal repeats in strings. We recall from [10, Section 7.12] that a *maximal repeat* is a substring occurring at least twice in a string or a set of strings and such that it cannot be extended to the left or to the right without losing occurrences.

Let $\mathbb{S} = s_1\$_1 s_2\$_2 \ldots s_k\$_k$, with the $\$_i$ being k different characters absent from the original alphabet Σ. The key point is that any maximal perfect haplotype block in S is a maximal repeat in $\mathbb{S}$. The opposite is not true: In a maximal perfect haplotype block, all occurrences of the repeat are located at the same position of each sequence of S (equality condition in Definition 1), while this constraint does not exist for maximal repeats in $\mathbb{S}$.

Nevertheless, finding all maximal perfect haplotype blocks in S can be performed by computing all maximal repeats in $\mathbb{S}$, while keeping only those whose occurrences are located at the same positions over all s_i in which they occur. This can be done by performing the following procedure[3]:

1 "Decorate" each sequence $s_i \in S$ to create $s_i^+ = \alpha_0 s_i[1]\alpha_1 s_i[2]\alpha_2 \ldots s_i[n]\alpha_n$, where the *index characters* $\alpha_0, \alpha_1, \ldots, \alpha_n$ are $n+1$ symbols from an alphabet Σ', disjoint from the original alphabet Σ.
2 Find in $\mathbb{S}^+ = s_1^+\$_1 s_2^+\$_2 \ldots s_k^+\$_k$ all maximal repeats.
3 Any maximal repeat $r = \alpha_p r_1 \alpha_{p+1} r_2 \alpha_{p+2} \ldots r_\ell \alpha_{p+\ell}$ in $\mathbb{S}^+$ with $\ell \geq 1$ corresponds to a maximal perfect haplotype block of length ℓ, starting at position $p+1$ in the input sequences from S.

The key idea here is that the index characters impose that each maximal repeat occurrence starts at the same position in all sequences and, as a consequence, ensure that all occurrences occur in distinct sequences from S.

Hence any maximal repeat $r = \alpha_p r_1 \alpha_{p+1} \ldots r_\ell \alpha_{p+\ell}$ defines a unique maximal perfect haplotype block $(K, p+1, p+\ell)$. The value $|K|$ is the number of occurrences of r. Also the set K can be derived from occurrence positions of r in $\mathbb{S}^+$, as any position in r corresponds to a unique position in $\mathbb{S}$. We prefer to omit useless technical details here.

The maximal repeat occurrences in $\mathbb{S}^+$ may be found using a suffix tree, constructed in time linear with respect to the size of the input data $O(kn)$, even for large integer alphabets [12], as we have here. The maximal repeat detection is also linear with the size of the input data [10, Section 7.12.1]. Therefore the overall time complexity is $O(kn)$.

Linear-time method II: based on the positional BWT

Here we present our second algorithm to find all maximal perfect haplotype blocks in linear time. It works by scanning the haplotype matrix column by column while maintaining the positional Burrows–Wheeler Transform (pBWT) [13] of the current column. For simplicity of presentation we assume that all rows of the haplotype matrix S are distinct. Recall that the pBWT of S consists of a pair of arrays for each column of S: For each l, $1 \leq l \leq n$, we have arrays a_l and d_l of length k such that the array a_l is a permutation of the elements in the set $\{1,2,\ldots,k\}$ with $S[a_l[1]][1,l] \leq \cdots \leq S[a_l[k]][1,l]$ colexicographically (i.e. right-to-left lexicographically) sorted, and the array d_l indicates the index from which the current and the previous rows coincide. Formally, $d_l[1] = l+1$ and for all r, $1 < r \leq k$, we have $d_l[r] = 1 + \max\{j \in [1,l] : S[a_l[r]][j] \neq S[a_l[r-1]][j]\}$. Further let us denote by a_l^{-1} the inverse permutation of a_l. For readers familiar with string processing terminology, the arrays a_l and a_l^{-1} are analogous to the suffix array and the inverse suffix array, respectively, while the arrays d_l are analogous to the LCP array.

[3] Note that a similar procedure has been described by Lunter [11], where also a connection to the positional Burrows–Wheeler Transform is mentioned.

1	2	3	4	5	6	a_6	a_6^{-1}
1	0	0	1	0	0	2	3
1	0	0	0	0	0	4	1
0	0	0	1	1	0	1	5
0	1	0	0	0	0	5	2
0	0	1	1	0	0	3	4

1	2	3	4	5	6	d_6
1	0	0	0	0	0	7
0	1	0	0	0	0	3
1	0	0	1	0	0	5
0	0	1	1	0	0	4
0	0	0	1	1	0	6

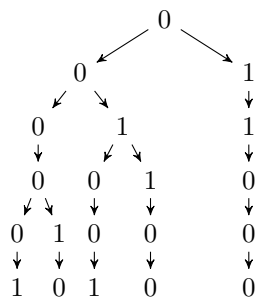

Fig. 2 Available blocks. *Left:* an example of a haplotype matrix up to column 6 with the two arrays a_6 and a_6^{-1} on the right. *Center:* the colexicographically sorted rows and the array d_6 listed on the right. *Right:* the trie of the reverses of the rows of the matrix. For example, the block $(\{1, 2, 4, 5\}, 5, 6)$ is available because $a_6^{-1}(1) = 3, a_6^{-1}(2) = 1, a_6^{-1}(4) = 2, a_6^{-1}(5) = 4$ is the consecutive range $[x, y] = [1, 4]$, we have $d_6[r] \leq 5$ for all $r \in [1 + 1, 4]$ with $d_6[3] = 5$, and we have $x = 1$ and $d_6[4 + 1] = 6 > 5$. The repeat in the block is 00, and we see it is a branching node in the trie on the right

Conditions 1, 2 and 4 (equality, left-maximality and row-maximality) of Definition 1 can be stated in terms of the arrays a_l and d_l as follows.

Definition 2 A quadruple $(i, j; x, y)$ with $1 \leq i \leq j \leq n$ and $1 \leq x < y \leq k$ is called an *available block* if the following holds:

- $d_j[r] \leq i$ for all $r \in [x + 1, y]$ (equality),
- there exists at least one $r \in [x + 1, y]$ such that $d_j[r] = i$ (left-maximality), and
- $(x = 1$ or $d_j[x] > i)$ and $(y = k$ or $d_j[y + 1] > i)$ (row-maximality).

The interval $[x, y]$ of an available block $(i, j; x, y)$ is called the *colexicographic range* of the block.

Lemma 1 *Suppose we have a maximal perfect haplotype block* (K, i, j)*, then the set* $\{a_j^{-1}[r] \mid r \in K\}$ *must be a contiguous range* $[x, y]$ *of indices such that* $(i, j; x, y)$ *is an available block.*

Proof This necessary condition follows immediately from Definitions 1 and 2 and the definition of the pBWT (arrays a_l and d_l). □

Let us consider the set B_l of available blocks ending at column l. We have that $|B_l| \leq k$ because each available block corresponds to a distinct branching node in the trie of the reverses of $\{S[1][1, l], \ldots, S[k][1, l]\}$, and the number of branching nodes in the trie is bounded from above by the number of leaves k. The branching nodes of the trie can be enumerated in $O(k)$ time by using a standard algorithm [14] for enumerating LCP intervals of the LCP array of the trie, $LCP_l[r] = l - d_l[r] + 1$. This gives us the colexicographic ranges $[x, y]$ of all available blocks in B_l. An example is shown in Fig. 2.

The only thing left is to show how to check the right-maximality property of an available block. The following lemma gives the sufficient condition for this:

Lemma 2 *An available block* $(i, j; x, y)$ *corresponds to a maximal haplotype block* (K, i, j) *if and only if* $j = n$ *or* $|\{S[a[r]][j + 1] : r \in [x, y]\}| > 1$.

Proof If $j = n$, right-maximality according to Definition 1 holds trivially. If $j < n$, right-maximality requires that there are two rows $s, t \in S|_K$ for which $s[j + 1] \neq t[j + 1]$. Since all rows s, t qualifying for this condition are within the colexicographic range $[x, y]$ of our available block, the statement follows immediately. □

To check the condition of Lemma 2 in constant time for $j \neq n$, we build a bit vector V_j such that $V_j[1] = 1$ and $V_j[r] = 1$ if and only if $S[a_j[r]][j + 1] \neq S[a_j[r - 1]][j + 1]$. Now the block is right-maximal if and only if $V_j[x + 1, y]$ contains at least one 1-bit. We can build a vector of prefix sums of V_j to answer this question in constant time.

Time and space complexity

We assume the *column stream model*, where we can stream the haplotype matrix column by column. We can thus build the arrays d_l, a_l and a_l^{-1} on the fly column by column [13], and also easily build the required prefix sums of arrays V_l from these. The time is $O(nk)$, since each of the n columns takes $O(k)$ time to process. The algorithm needs to keep in memory only the data for two adjacent columns at a time, so in space $O(k)$ we can report the colexicographic ranges of all maximal blocks ending in each column $l \in [1, n]$. If the colexicographic range of

Table 1 Running times and memory usage of our pBWT-based implementation

Data set	#lines	#columns	Min block size	Time	Memory (MB)	#blocks
chr. 22	5008	1,055,454		4 min 54 s	12.8	148,613,645
chr. 22	5008	1,055,454	500,000	3 min 50 s	12.8	16,076,453
chr. 22	5008	1,055,454	1,000,000	3 min 40 s	12.8	2,228,762
chr. 22	5008	1,055,454	2,000,000	3 min 43 s	12.8	4779
chr. 6	5008	4,800,101		19 min 42 s	12.8	624,689,548
chr. 6	5008	4,800,101	500,000	17 min 20 s	12.8	89,840,467
chr. 6	5008	4,800,101	1,000,000	16 min 30 s	12.8	11,388,982
chr. 6	5008	4,800,101	2,000,000	16 min 36 s	12.8	5585
chr. 2	5008	6,786,300		31 min 57 s	12.8	946,717,897
chr. 2	5008	6,786,300	500,000	25 min 06 s	12.8	160,094,115
chr. 2	5008	6,786,300	1,000,000	23 min 24 s	12.8	25,533,314
chr. 2	5008	6,786,300	2,000,000	23 min 18 s	12.8	120,243

Note that in our streaming implementation the memory usage is dominated by the number of haplotypes times the buffer size, and therefore is essentially constant in this study

a block at column l is $[x, y]$, then the rows in the original haplotype matrix are $a_l[x], a_l[x+1], \ldots, a_l[y]$. There are $O(nk)$ blocks and $O(k)$ rows per block, so the time to report all rows explicitly is $O(nk^2)$. In fact, a sharper bound that can also easily be achieved is $O(nk + z)$ where $z \in O(nk^2)$ is the size of the output. Alternatively, we can store a complete representation of the answer taking $O(nk)$ space by storing all the a_l arrays and the colexicographic ranges of the maximal perfect blocks for each column, from which we can readily report all rows in any maximal perfect block in constant time per row.

Empirical evaluation

Since the algorithm of "Linear-time method I: based on suffix trees" section is mostly of theoretical interest, we evaluate only the pBWT-based algorithm presented in "Linear-time method II: based on the positional BWT" section. The source code is available from https://gitlab.com/bacazaux/haploblocks. As a baseline for comparison we use the implementation of the trie-based algorithm by Cunha et al. [8], available from the same gitlab site. The experiments were run on a machine with an Intel Xeon E5-2680 v4 2.4 GHz CPU, which has a 35 MB Intel SmartCache. The machine has 256 gigabytes of memory at a speed of 2400MT/s. The code was compiled with `g++` using the `-Ofast` optimization flag.

Our test data consists of chromosomes 2, 6 and 22 from phase three of the 1000 Genomes Project [2], which provides whole-genome sequences of 2504 individuals from multiple populations worldwide. We preprocessed the data by extracting all biallelic SNPs from the provided VCF files[4] and converting them to a binary haplotype matrix using our own program `vcf2bm`, also available from https://gitlab.com/bacazaux/haploblocks.

Table 2 Comparison of the trie-based implementation from [8] and our pBWT-based implementation with minimum block size 10^6

Data set	`trie`		`pBWT`	
	Time	Memory	Time	Memory (MB)
chr. 22	17 min 08 s	927.8 MB	3 min 40 s	12.8
chr. 6	1 h 34 min 34 s	3.23 GB	16 min 30 s	12.8
chr. 2	2 h 07 min 21 s	4.46 GB	23 min 24 s	12.8

Our implementation has a user-defined parameter allowing to adjust the minimum size of a reported maximal perfect haplotype block (K, i, j), where *size* is defined as the width $(j - i + 1)$ times the number of rows $(|K|)$ in the block. Table 1 shows the running times and memory usage of our implementation on the different chromosomes and for different settings of the minimum block size parameter. The larger the minimum block size, the faster the algorithm is, because there are less blocks to report. In general, it takes only a few minutes to process a complete human chromosome. Locating all 323,163,970 blocks of minimum size 10^6 in all 22 human autosomes (non-sex chromosomes) took in total 4 h and 26 min with a memory peak of 12.8 MB (data not shown).

Table 2 shows a comparison of our implementation to the trie-based implementation from [8]. Our implementation is about 5 times faster on all datasets, and the memory consumption is up to 93 times smaller.

It is now easy to apply the method for estimating a local selection coefficient from the size of maximal perfect haplotype blocks covering a certain genomic region

[4] ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/.

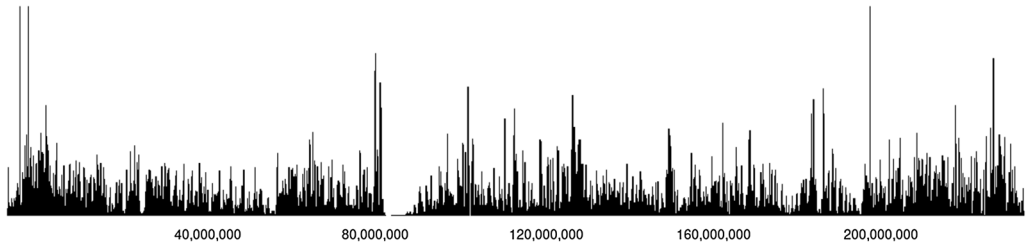

Fig. 3 Selection scan for human chromosome 2. Shown is for each position of the chromosome the largest maximum likelihood estimate derived from any maximal perfect haplotype block overlapping that locus. It is easy to spot potential regions of high selection. The centromere, located around 93 Mbp, shows no signal as sequencing coverage is low here and no SNPs could be called

presented in [8]. This method estimates the likelihood of observing a haplotype block for a given selection coefficient s and the time t since the onset of selection following an approach presented by Chen et al. [15]. Therefore, chromosome-wide selection scans indicating the loci of maximum selection, as shown in Fig. 3 for the complete human chromosome 2 (size parameter 10^6), can now be generated in less than half an hour.

Conclusion

In this paper we presented two algorithms that are able to find all maximal perfect haplotype blocks in a haplotype matrix of size $k \times n$ in linear time $O(kn)$. In particular the second method, based on the positional Burrows–Wheeler Transform, performs also extremely well in practice, as it allows for a streaming implementation with extremely low memory footprint.

While an initial implementation of the method is available from https://gitlab.com/bacazaux/haploblocks, a user-friendly software combining the algorithm presented here with the computation of the selection coefficient suggested in [8] remains to be developed.

Acknowledgements
We thank the organizers of DSB 2019 (http://www.dsb2019.gitlab.io) for giving us the opportunity to present earlier work in this area and start a discussion from which the present results originated. We would also like to thank Michel T. Henrichs for providing a script to convert VCF files to haplotype matrices and for assisting with the production of Fig. 3.

Authors' contributions
All authors contributed equally. All authors read and approved the final manuscript.

Funding
The work presented in this article and the publication cost were funded from the authors' home institutions.
This funding was supported by JSPS KASKENHI (Grant No JP16H02783); ANR Hydrogen (Grant No ANR-14-CE23-0001).

Availability of data and materials
Source code and test data are available from https://gitlab.com/bacazaux/haploblocks.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Department of Computer Science, University of Helsinki, Helsinki, Finland. [2] Department of Informatics, Kyushu University, Fukuoka, Japan. [3] Inria, CNRS, Irisa, Univ. Rennes, Rennes, France. [4] Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Bielefeld, Germany.

Received: 18 October 2019 Accepted: 28 January 2020
Published online: 10 February 2020

References

1. Buniello A, MacArthur JAL, Cerezo M, Harris LW, Hayhurst J, Malangone C, McMahon A, Morales J, Mountjoy E, Sollis E, Suveges D, Vrousgou O, Whetzel PL, Amode R, Guillen JA, Riat HS, Trevanion SJ, Hall P, Junkins H, Flicek P, Burdett T, Hindorff LA, Cunningham F, Parkinson H. The NHGRI-EBI GWAS catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. Nucl Acids Res. 2018;47(D1):1005–12. https://doi.org/10.1093/nar/gky1120.
2. Auton A, Brooks LD, Durbin RM, Garrison EP, Min Kang H, Korbel JO, Marchini JL, McCarthy S, McVean GA, Abecasis GR, 1000 Genomes Project Consortium. A global reference for human genetic variation. Nature. 2015;526(7571):68–74. https://doi.org/10.1038/nature15393.
3. Sudmant PH, Rausch T, Gardner EJ, Handsaker RE, Abyzov A, Huddleston J, Zhang Y, Ye K, Jun G, Hsi-Yang Fritz M, Konkel MK, Malhotra A, Stütz AM, Shi X, Paolo Casale F, Chen J, Hormozdiari F, Dayama G, Chen K, Malig M, Chaisson MJP, Walter K, Meiers S, Kashin S, Garrison E, Auton A, Lam HYK, Jasmine MuX, Alkan C, Antaki D, Bae T, Cerveira E, Chines P, Chong Z, Clarke L, Dal E, Ding L, Emery S, Fan X, Gujral M, Kahveci F, Kidd JM, Kong Y, Lameijer E-W, McCarthy S, Flicek P, Gibbs RA, Marth G, Mason CE, Menelaou A, Muzny DM, Nelson BJ, Noor A, Parrish NF, Pendleton M, Quitadamo A, Raeder B, Schadt EE, Romanovitch M, Schlattl A, Sebra R, Shabalin AA, Untergasser A, Walker JA, Wang M, Yu F, Zhang C, Zhang J, Zheng-Bradley X, Zhou W, Zichner T, Sebat J, Batzer MA, McCarroll SA, Mills RE, Gerstein MB, Bashir A, Stegle O, Devine SE, Lee C, Eichler EE, Korbel JO, The 1000 Genomes Project Consortium. An integrated map of structural variation in 2504 human genomes. Nature. 2015;526(7571):75–81. https://doi.org/10.1038/nature15394.
4. Gudbjartsson DF, Helgason H, Gudjonsson SA, Zink F, Oddson A, Gylfason A, Besenbacher S, Magnusson G, Halldorsson BV, Hjartarson E, Sigurdsson GT, Stacey SN, Frigge ML, Holm H, Saemundsdottir J, Helgadottir HT, Johannsdottir H, Sigfusson G, Thorgeirsson G, Sverrisson JT, Gretarsdottir S, Walters GB, Rafnar T, Thjodleifsson B, Bjornsson ES, Olafsson S,

Thorarinsdottir H, Steingrimsdottir T, Gudmundsdottir TS, Theodors A, Jonasson JG, Sigurdsson A, Bjornsdottir G, Jonsson JJ, Thorarensen O, Ludvigsson P, Gudbjartsson H, Eyjolfsson GI, Sigurdardottir O, Olafsson I, Arnar DO, Magnusson OT, Kong A, Masson G, Thorsteinsdottir U, Helgason A, Sulem P, Stefansson K. Large-scale whole-genome sequencing of the Icelandic population. Nat Genet. 2015;47:435–44. https://doi.org/10.1038/ng.3247.

5. Hehir-Kwa JY, Marschall T, Kloosterman WP, Francioli LC, Baaijens JA, Dijkstra LJ, Abdellaoui A, Koval V, Thung DT, Wardenaar R, Renkens I, Coe BP, Deelen P, de Ligt J, Lameijer E-W, van Dijk F, Hormozdiari F, Consortium TGotN, Bovenberg JA, de Craen AJM, Beekman M, Hofman A, Willemsen G, Wolffenbuttel B, Platteel M, Du Y, Chen R, Cao H, Cao R, Sun Y, Cao JS, Neerincx PBT, Dijkstra M, Byelas G, Kanterakis A, Bot J, Vermaat M, Laros JFJ, den Dunnen JT, de Knijff P, Karssen LC, van Leeuwen EM, Amin N, Rivadeneira F, Estrada K, Hottenga J-J, Kattenberg VM, van Enckevort D, Mei H, Santcroos M, van Schaik BDC, Handsaker RE, McCarroll SA, Ko A, Sudmant P, Nijman IJ, Uitterlinden AG, van Duijn CM, Eichler EE, de Bakker PIW, Swertz MA, Wijmenga C, van Ommen G-JB, Slagboom PE, Boomsma DI, Schönhuth A, Ye K, Guryev V. A high-quality human reference panel reveals the complexity and distribution of genomic structural variants. Nat Commun. 2016;7:12989. https://doi.org/10.1038/ncomms12989.
6. Turnbull C, Scott RH, Thomas E, Jones L, Murugaesu N, Pretty FB, Halai D, Baple E, Craig C, Hamblin A, Henderson S, Patch C, O'Neill A, Devereau A, Smith K, Martin AR, Sosinsky A, McDonagh EM, Sultana R, Mueller M, Smedley D, Toms A, Dinh L, Fowler T, Bale M, Hubbard TJP, Rendon A, Hill S, Caulfield MJ. 100 000 Genomes Project: the 100 000 genomes project: bringing whole genome sequencing to the NHS. BMJ. 2018;361:1687. https://doi.org/10.1136/bmj.k1687.
7. Gillespie JH. Population genetics—a concise guide. Baltimore: The Johns Hopkins University Press; 1998.
8. Cunha L, Diekmann Y, Kowada LAB, Stoye J Identifying maximal perfect haplotype blocks. In: Advances in bioinformatics and computational biology: 11th Brazilian symposium on bioinformatics, BSB 2018, Niterói, Brazil, October 30 - November 1, 2018, Proceedings; 2018. p. 26–37. https://doi.org/10.1007/978-3-030-01722-4_3.
9. Alanko J, Bannai H, Cazaux B, Peterlongo P, Stoye J Finding all maximal perfect haplotype blocks in linear time. In: Huber, K.T., Gusfield, D. (eds.) 19th International Workshop on Algorithms in Bioinformatics (WABI 2019). LIPIcs, vol. 143:8, p. 1–9 (2019). https://doi.org/10.4230/LIPIcs.WABI.2019.8
10. Gusfield D. Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge: Cambridge University Press; 1997.
11. Lunter G. Haplotype matching in large cohorts using the Li and Stephens model. Bioinformatics. 2019;35(5):798–806. https://doi.org/10.1093/bioinformatics/bty735.
12. Farach M Optimal suffix tree construction with large alphabets. In: Proceedings 38th annual symposium on foundations of computer science. New York: IEEE; 1997. p. 137–143.
13. Durbin R. Efficient haplotype matching and storage using the positional Burrows–Wheeler transform (PBWT). Bioinformatics. 2014;30(9):1266–72. https://doi.org/10.1093/bioinformatics/btu014.
14. Abouelhoda MI, Kurtz S, Ohlebusch E. Replacing suffix trees with enhanced suffix arrays. J Discret Algorithms. 2004;2(1):53–86. https://doi.org/10.1016/S1570-8667(03)00065-0.
15. Chen H, Hey J, Slatkin M. A hidden Markov model for investigating recent positive selection through haplotype structure. Theor Popul Biol. 2015;99:18–30. https://doi.org/10.1016/j.tpb.2014.11.001.

Publisher's Note